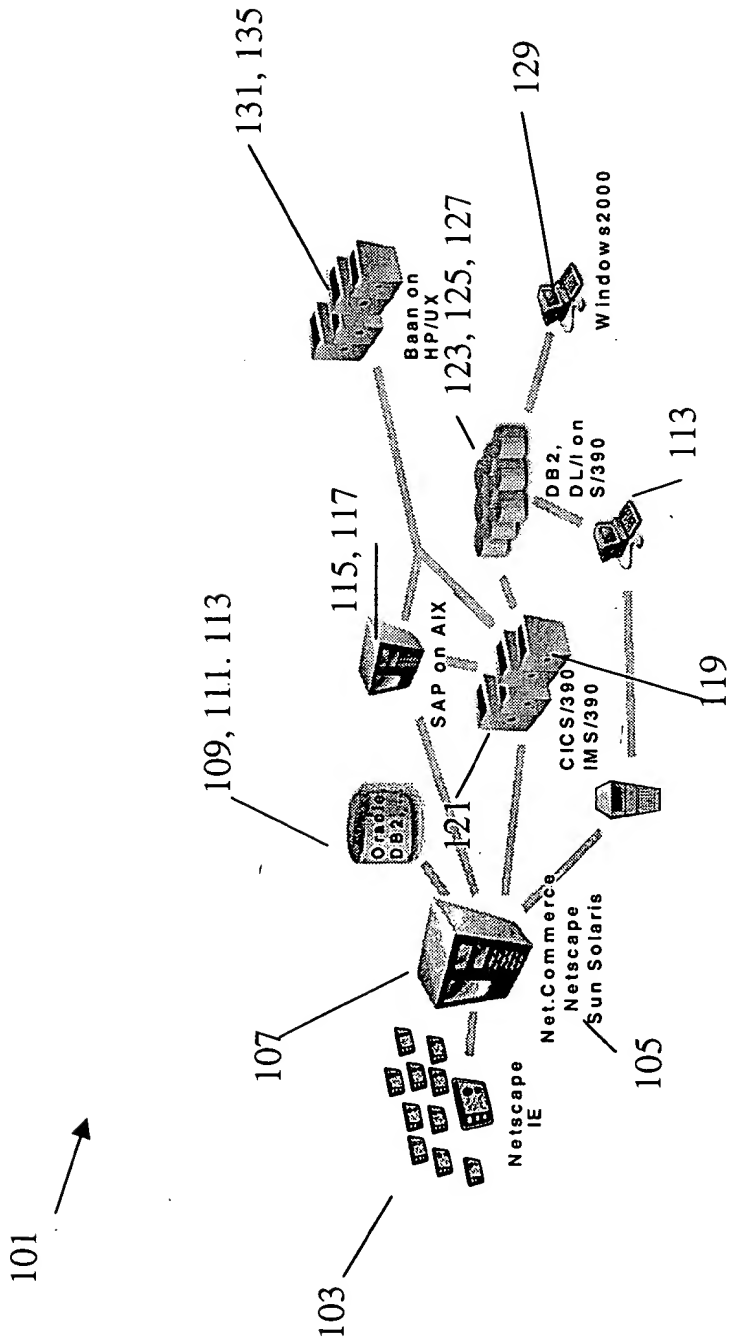
101
103
107
105
109, 111. 113
115, 117
121
119
123, 125, 127
129
131, 135
Netscape IE
Net.Commerce Netscape Sun Solaris
Oracle DB2
SAP on AIX
CICS/390 IMS/390
DB2, DL/I on S/390
Baan on HP/UX
Windows2000

Figure1

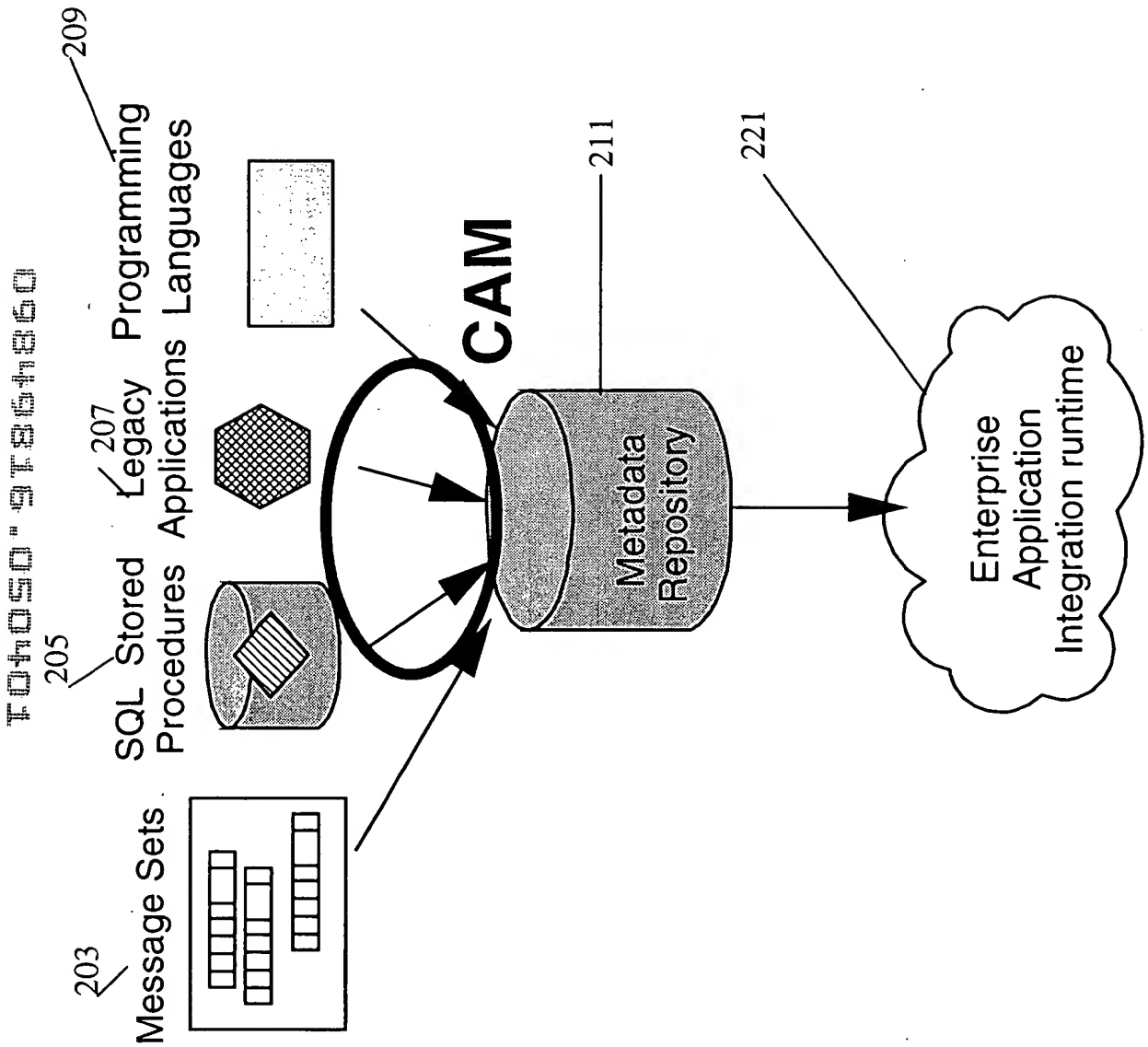
203
Message Sets
205
SQL Stored Procedures
207
Legacy Applications
209
Programming Languages
CAM
211
Metadata Repository
221
Enterprise Application Integration runtime

Figure 2

FOHO50 9T86H860

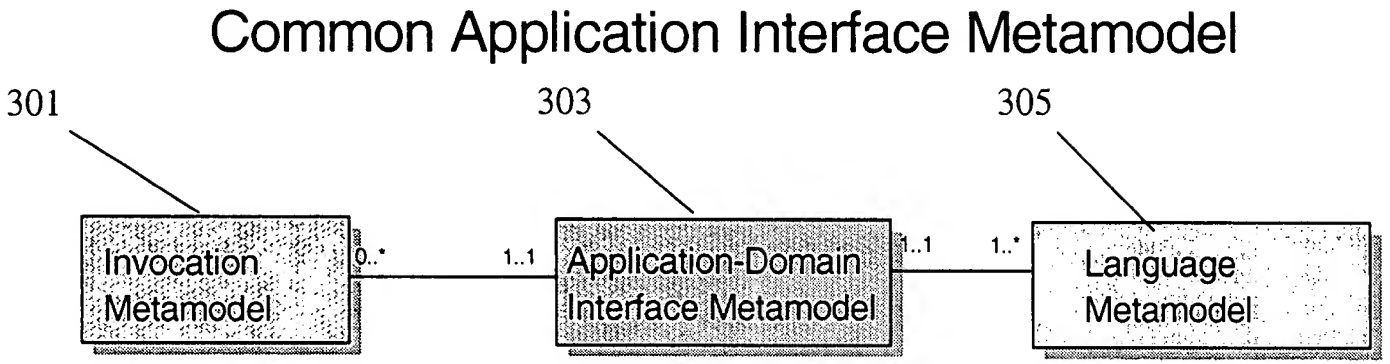

Figure 3

TO:050" 9T86:860

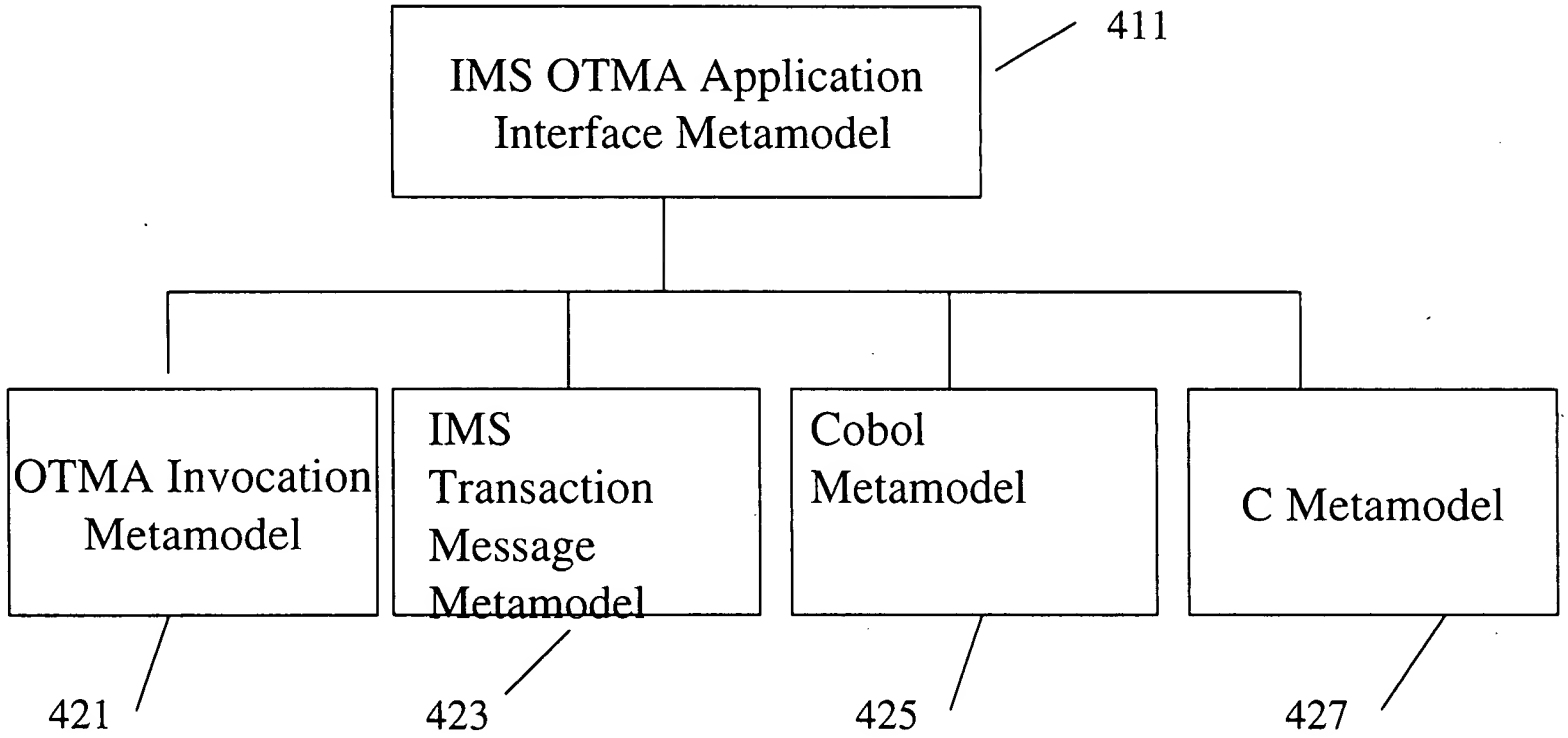

Figure 4

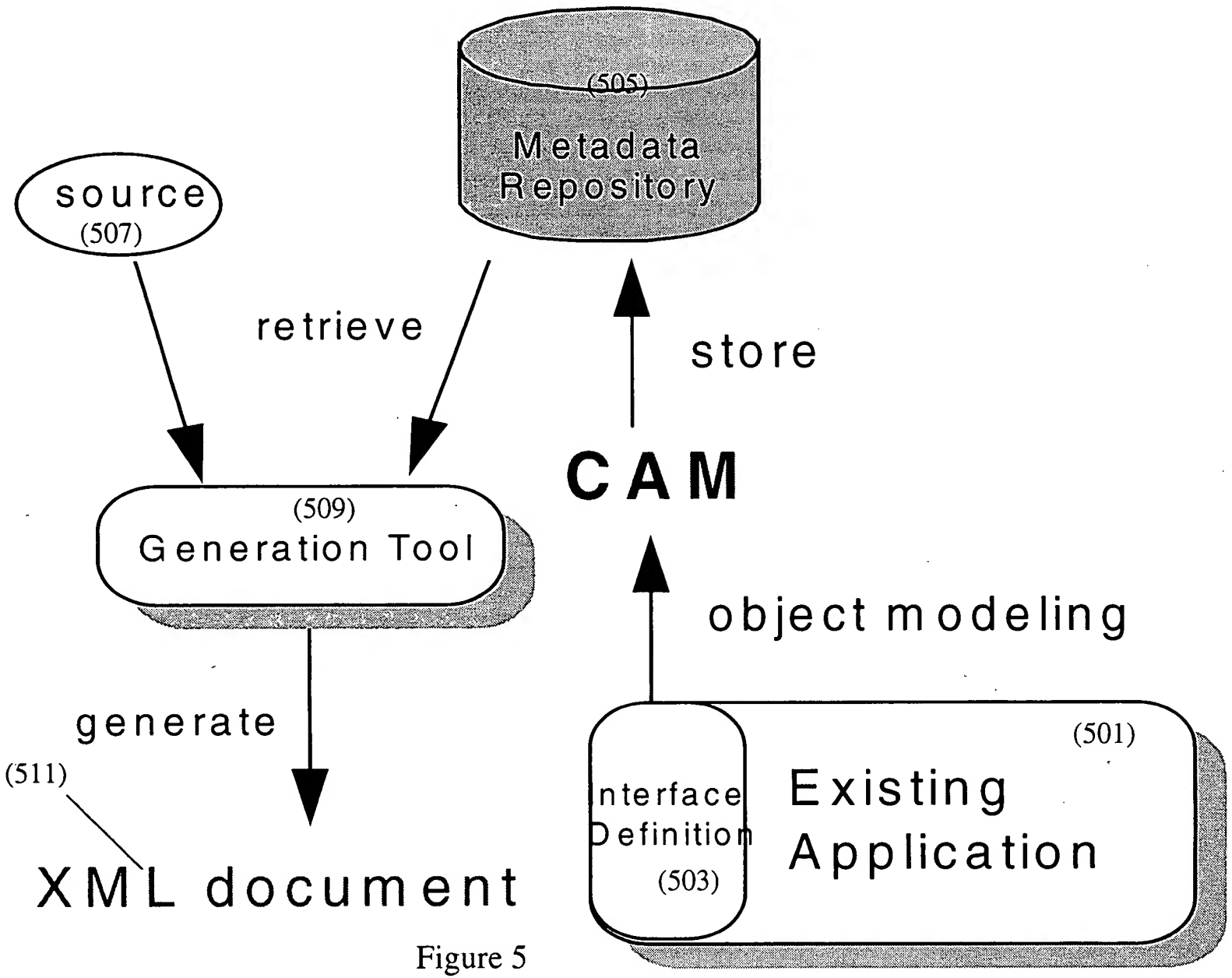
(505)
Metadata Repository
source
(507)
retrieve
store
CAM
(509)
Generation Tool
object modeling
generate
(511)
XML document
(501)
Interface Definition
(503)
Existing Application

Figure 5

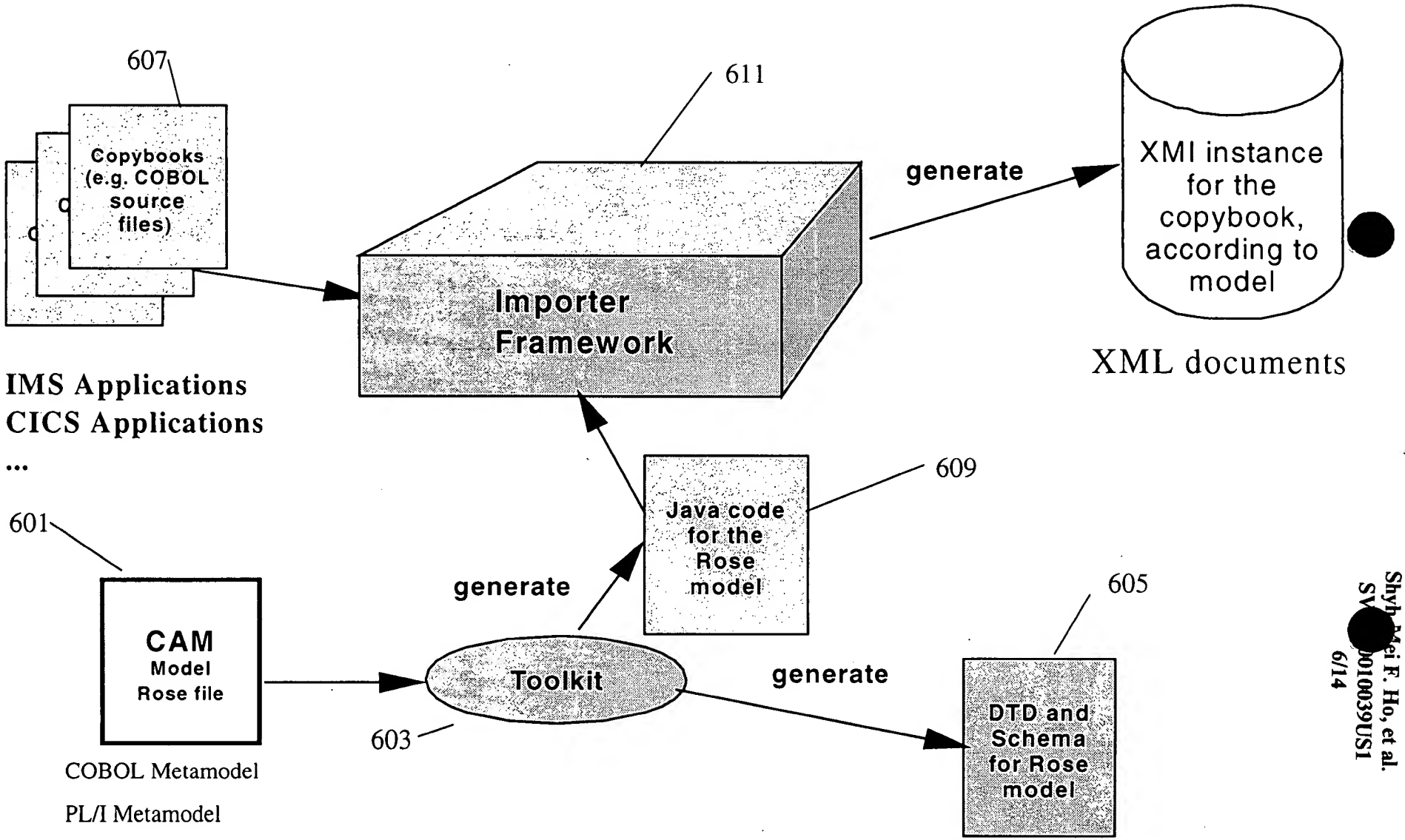
607
Copybooks (e.g. COBOL source files)
611
Importer Framework
generate
XMI instance for the copybook, according to model
XML documents
IMS Applications
CICS Applications
...
601
CAM
Model
Rose file
generate
Java code for the Rose model
609
605
Toolkit
603
generate
DTD and Schema for Rose model
COBOL Metamodel
PL/I Metamodel
...
MFS Metamodel
BMS Metamodel

Figure 6

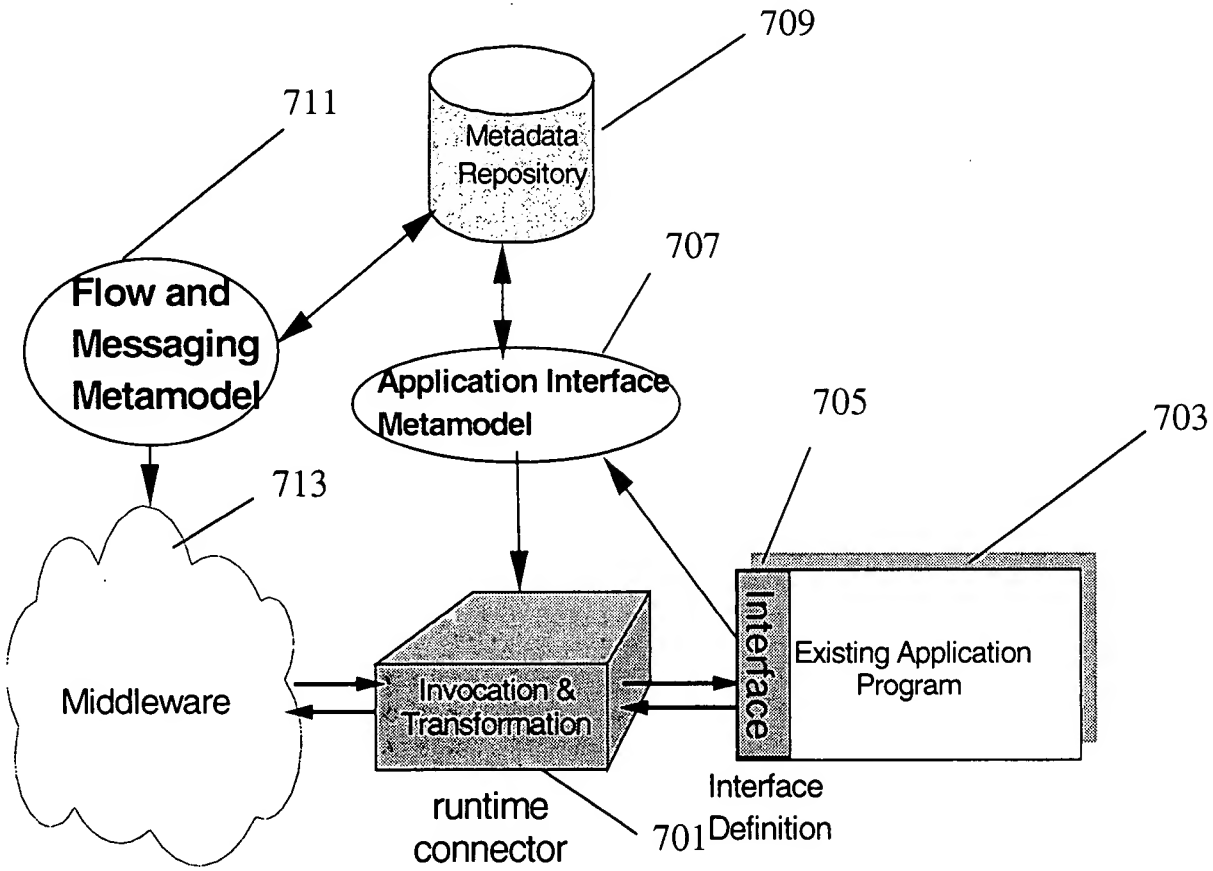
709
711
Metadata Repository
707
Flow and Messaging Metamodel
Application Interface Metamodel
705
703
713
Middleware
Invocation & Transformation
Interface
Existing Application Program
runtime connector
701
Interface Definition

Figure 7

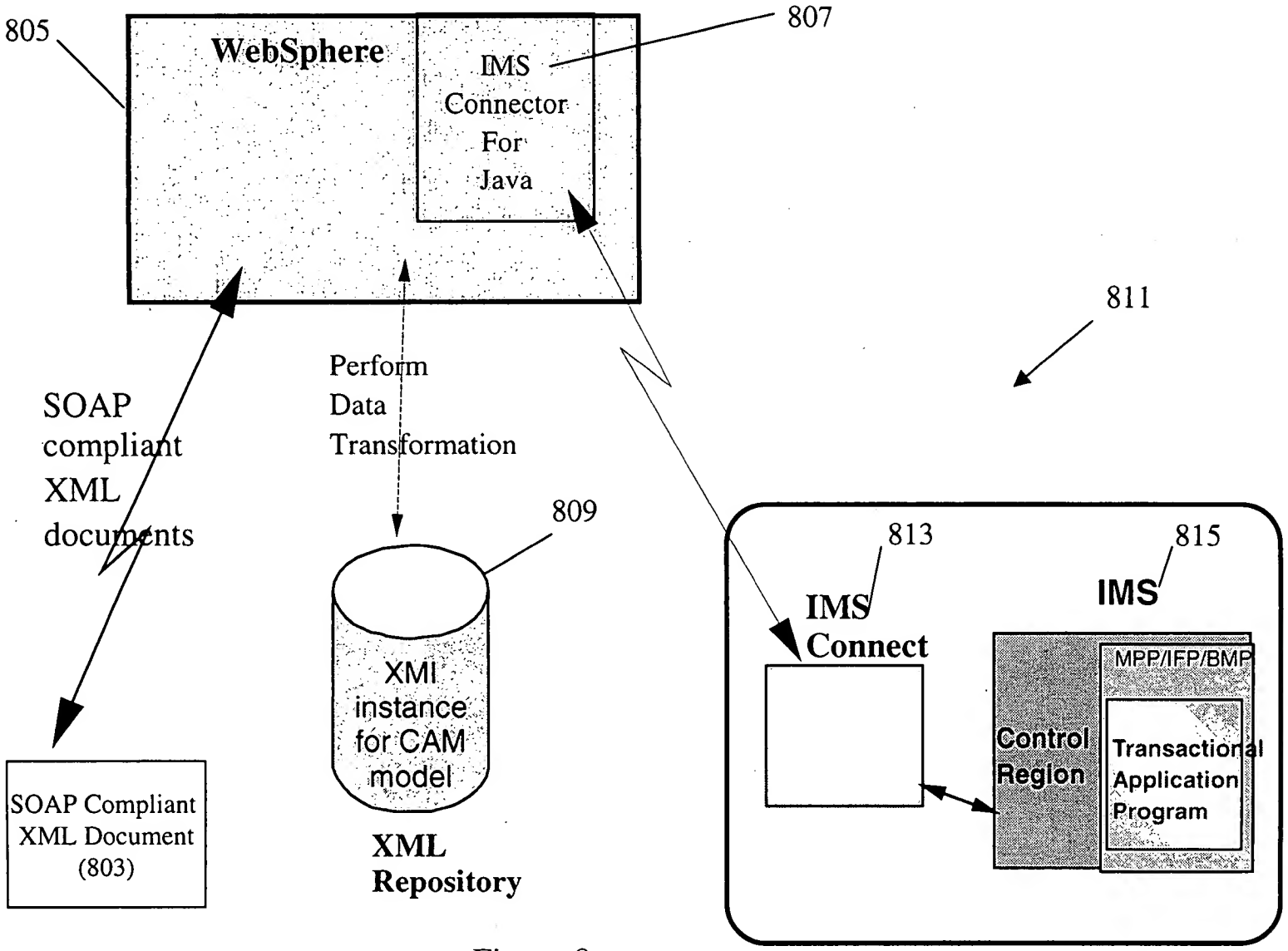
805
WebSphere
807
IMS
Connector
For
Java
811
Perform
Data
Transformation
SOAP
compliant
XML
documents
809
813
815
IMS
IMS
Connect
MPP/IFP/BMP
Control
Region
Transactional
Application
Program
SOAP Compliant
XML Document
(803)
XMI
instance
for CAM
model
XML
Repository

Figure 8

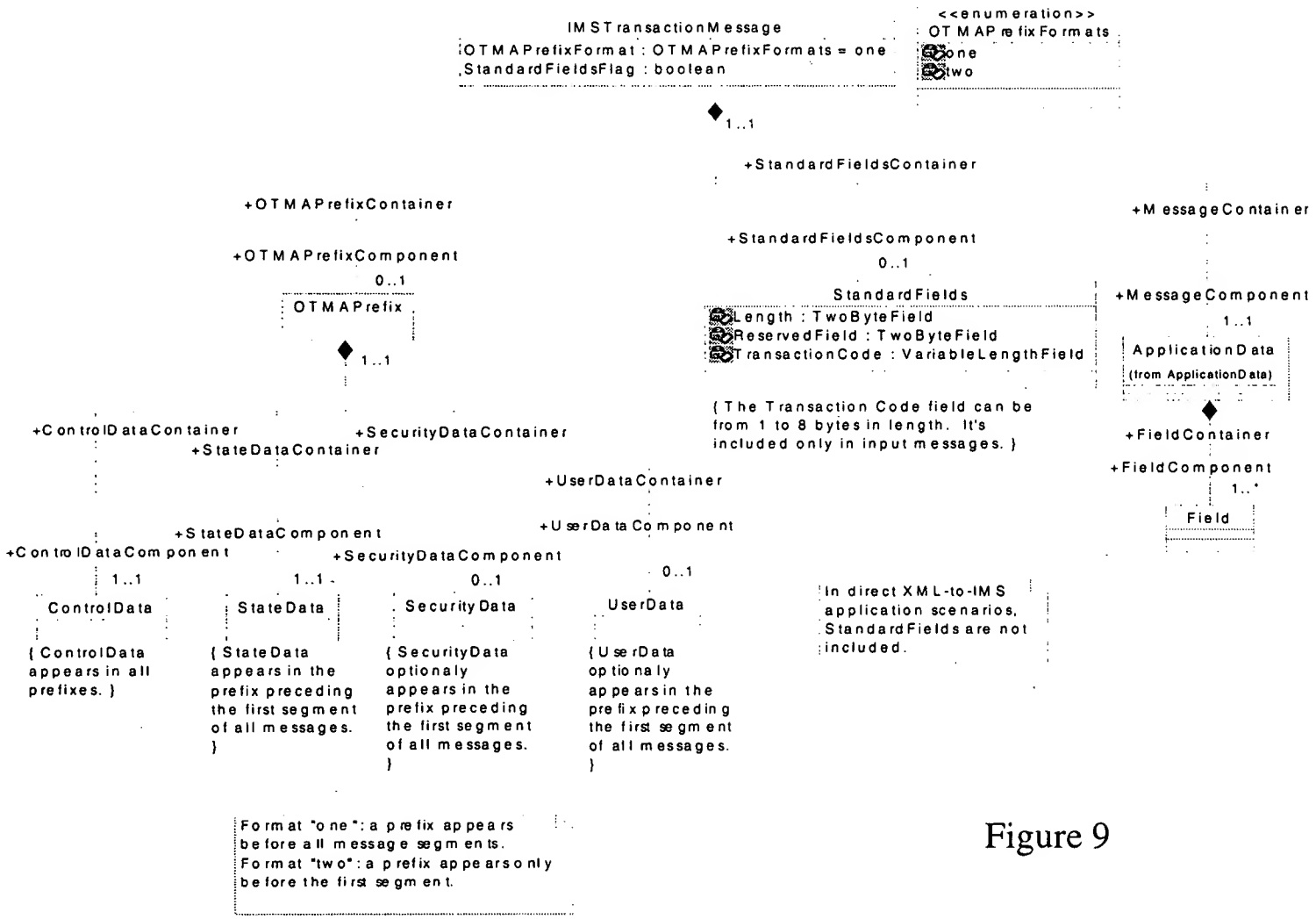
IMSTransactionMessage
OTMAPrefixFormat : OTMAPrefixFormats = one
StandardFieldsFlag : boolean
<<enumeration>>
OTMAPrefixFormats
one
two
1..1
+StandardFieldsContainer
+StandardFieldsComponent
0..1
StandardFields
Length : TwoByteField
ReservedField : TwoByteField
TransactionCode : VariableLengthField
{ The Transaction Code field can be from 1 to 8 bytes in length. It's included only in input messages. }
+OTMAPrefixContainer
+OTMAPrefixComponent
0..1
OTMAPrefix
1..1
+ControlDataContainer
+StateDataContainer
+SecurityDataContainer
+UserDataContainer
+UserDataComponent
+StateDataComponent
+ControlDataComponent
+SecurityDataComponent
1..1
1..1
0..1
0..1
ControlData
StateData
SecurityData
UserData
{ ControlData appears in all prefixes. }
{ StateData appears in the prefix preceding the first segment of all messages. }
{ SecurityData optionaly appears in the prefix preceding the first segment of all messages. }
{ UserData optionaly appears in the prefix preceding the first segment of all messages. }
Format "one": a prefix appears before all message segments.
Format "two": a prefix appears only before the first segment.
+MessageContainer
+MessageComponent
1..1
ApplicationData
(from ApplicationData)
+FieldContainer
+FieldComponent
1..*
Field
In direct XML-to-IMS application scenarios, StandardFields are not included.

Figure 9

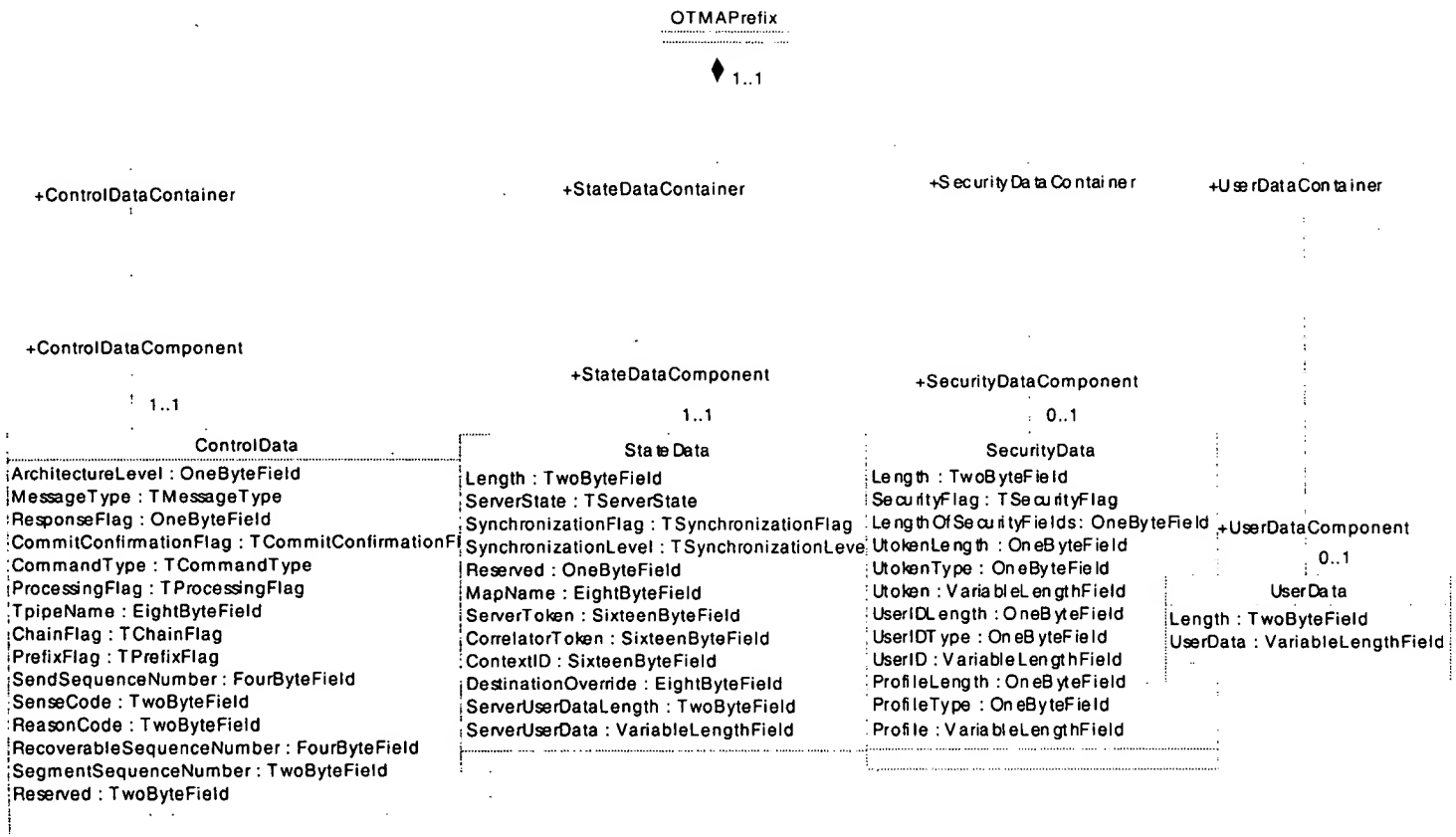
OTMAPrefix
1..1
+ControlDataContainer
+StateDataContainer
+SecurityDataContainer
+UserDataContainer
+ControlDataComponent
1..1
ControlData
ArchitectureLevel : OneByteField
MessageType : TMessageType
ResponseFlag : OneByteField
CommitConfirmationFlag : TCommitConfirmationFl
CommandType : TCommandType
ProcessingFlag : TProcessingFlag
TpipeName : EightByteField
ChainFlag : TChainFlag
PrefixFlag : TPrefixFlag
SendSequenceNumber : FourByteField
SenseCode : TwoByteField
ReasonCode : TwoByteField
RecoverableSequenceNumber : FourByteField
SegmentSequenceNumber : TwoByteField
Reserved : TwoByteField
+StateDataComponent
1..1
StateData
Length : TwoByteField
ServerState : TServerState
SynchronizationFlag : TSynchronizationFlag
SynchronizationLevel : TSynchronizationLeve
Reserved : OneByteField
MapName : EightByteField
ServerToken : SixteenByteField
CorrelatorToken : SixteenByteField
ContextID : SixteenByteField
DestinationOverride : EightByteField
ServerUserDataLength : TwoByteField
ServerUserData : VariableLengthField
+SecurityDataComponent
0..1
SecurityData
Length : TwoByteField
SecurityFlag : TSecurityFlag
LengthOfSecurityFields: OneByteField
UtokenLength : OneByteField
UtokenType : OneByteField
Utoken : VariableLengthField
UserIDLength : OneByteField
UserIDType : OneByteField
UserID : VariableLengthField
ProfileLength : OneByteField
ProfileType : OneByteField
Profile : VariableLengthField
+UserDataComponent
0..1
UserData
Length : TwoByteField
UserData : VariableLengthField

Figure 10

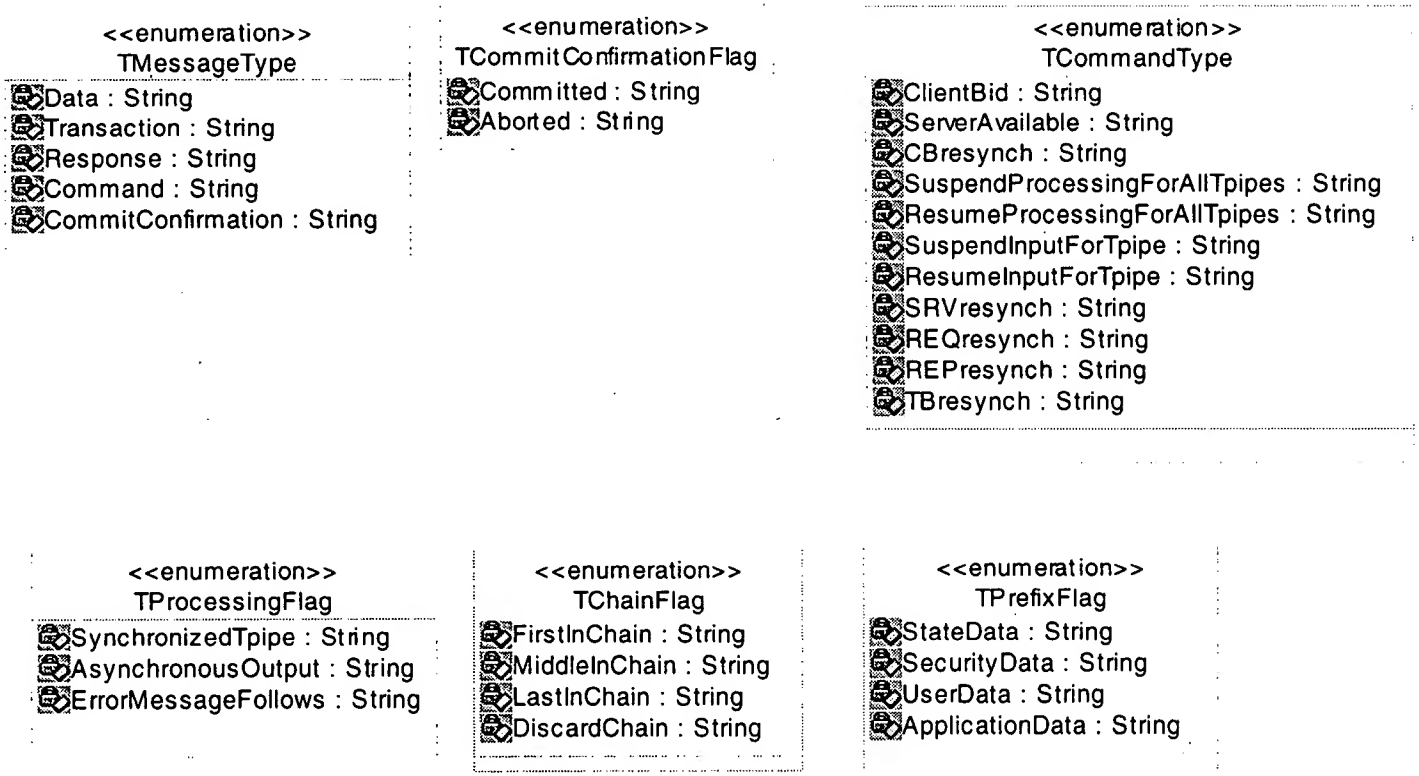
<<enumeration>>
TMessageType
Data : String
Transaction : String
Response : String
Command : String
CommitConfirmation : String
<<enumeration>>
TCommitConfirmationFlag
Committed : String
Aborted : String
<<enumeration>>
TCommandType
ClientBid : String
ServerAvailable : String
CBresynch : String
SuspendProcessingForAllTpipes : String
ResumeProcessingForAllTpipes : String
SuspendInputForTpipe : String
ResumeInputForTpipe : String
SRVresynch : String
REQresynch : String
REPresynch : String
TBresynch : String
<<enumeration>>
TProcessingFlag
SynchronizedTpipe : String
AsynchronousOutput : String
ErrorMessageFollows : String
<<enumeration>>
TChainFlag
FirstInChain : String
MiddleInChain : String
LastInChain : String
DiscardChain : String
<<enumeration>>
TPrefixFlag
StateData : String
SecurityData : String
UserData : String
ApplicationData : String

Figure 11

T0h050" 9T86h860

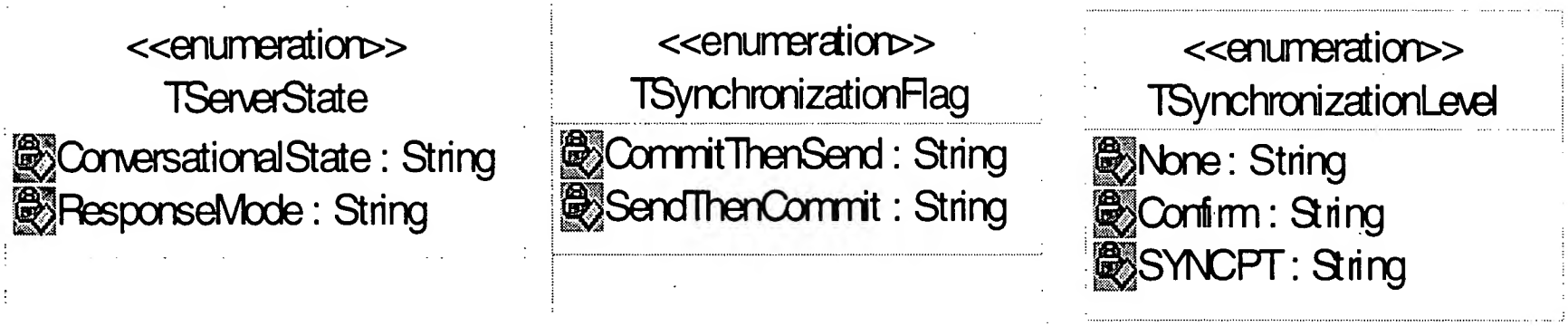

Figure 12

T0h050" 9T86h860

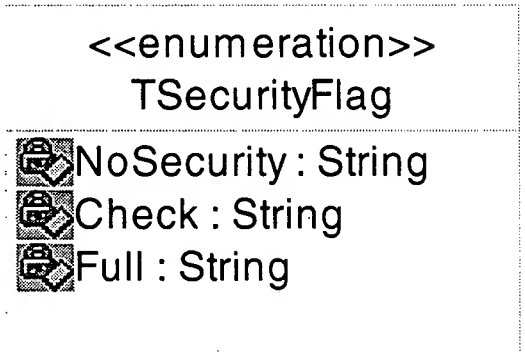

Figure 13

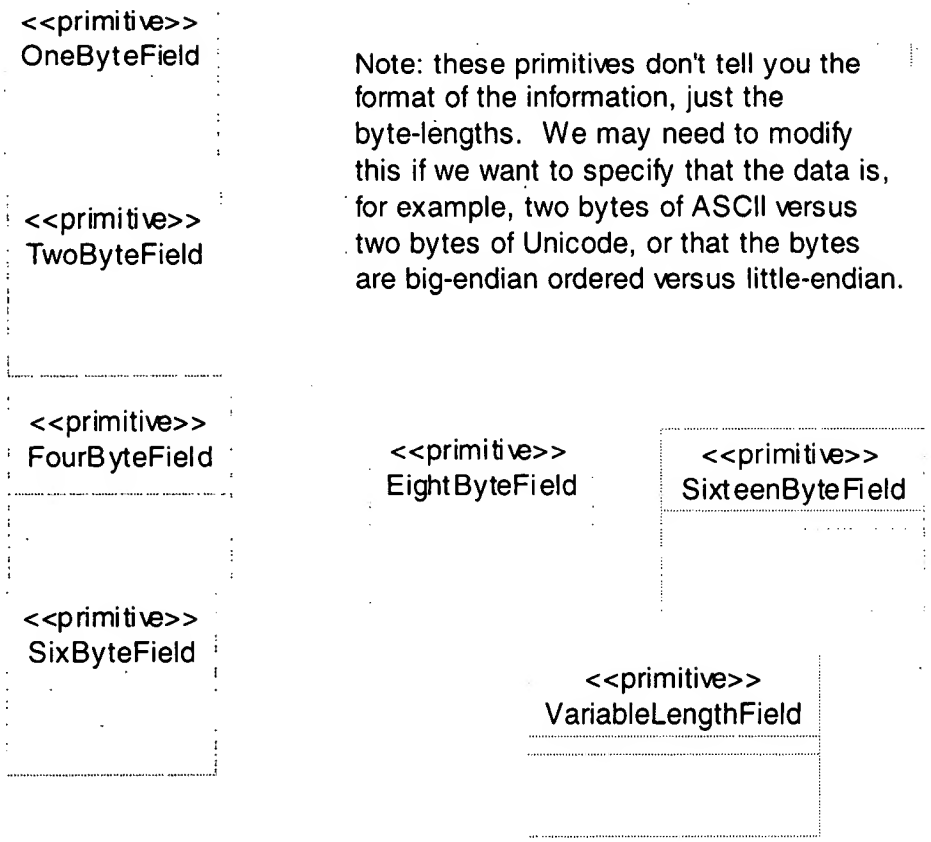
<<primitive>>
OneByteField
<<primitive>>
TwoByteField
<<primitive>>
FourByteField
<<primitive>>
SixByteField
Note: these primitives don't tell you the format of the information, just the byte-lengths. We may need to modify this if we want to specify that the data is, for example, two bytes of ASCII versus two bytes of Unicode, or that the bytes are big-endian ordered versus little-endian.
<<primitive>>
EightByteField
<<primitive>>
SixteenByteField
<<primitive>>
VariableLengthField

Figure 14